Verso l'architettura MVC-2
i JavaBeans

ALBERTO BELUSSI

ANNO ACCADEMICO 2012/2013

Limiti dell'approccio PROGRAMMA UNICO

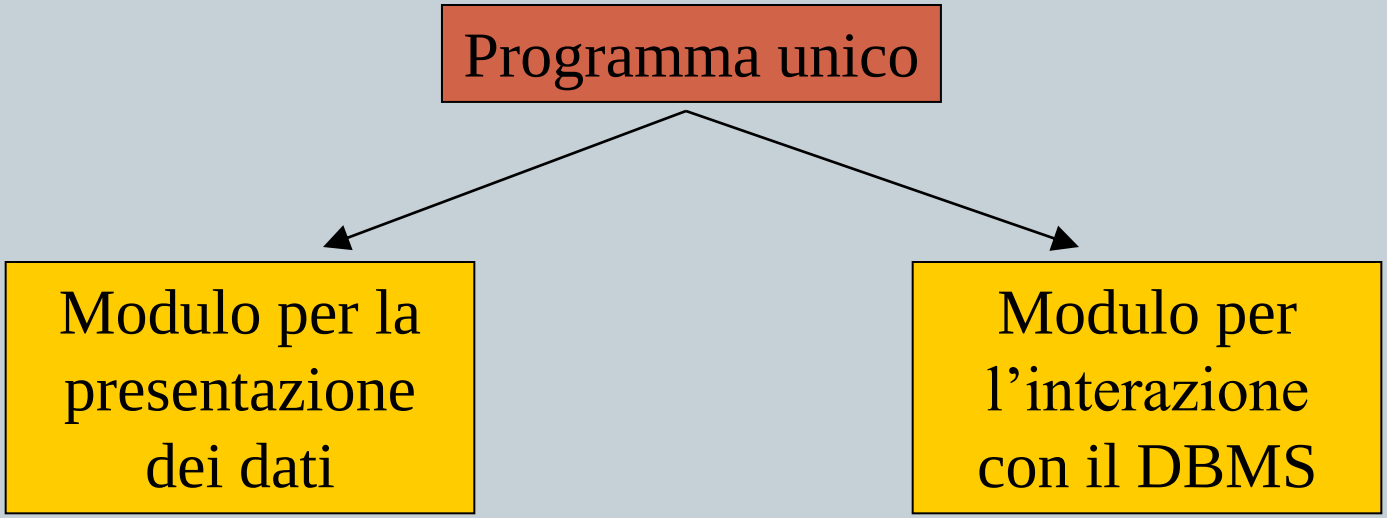
Il programma unico (servlet) svolge tre tipi di funzioni distinte:

- Gestione della richiesta HTTP
- Interazione con il DBMS ed elaborazione dei dati estratti
- Presentazione dei dati elaborati nella pagina web scritta in HTML (struttura dell'informazione e aspetti grafici)

Tale situazione implica un costo elevato di manutenzione del codice.

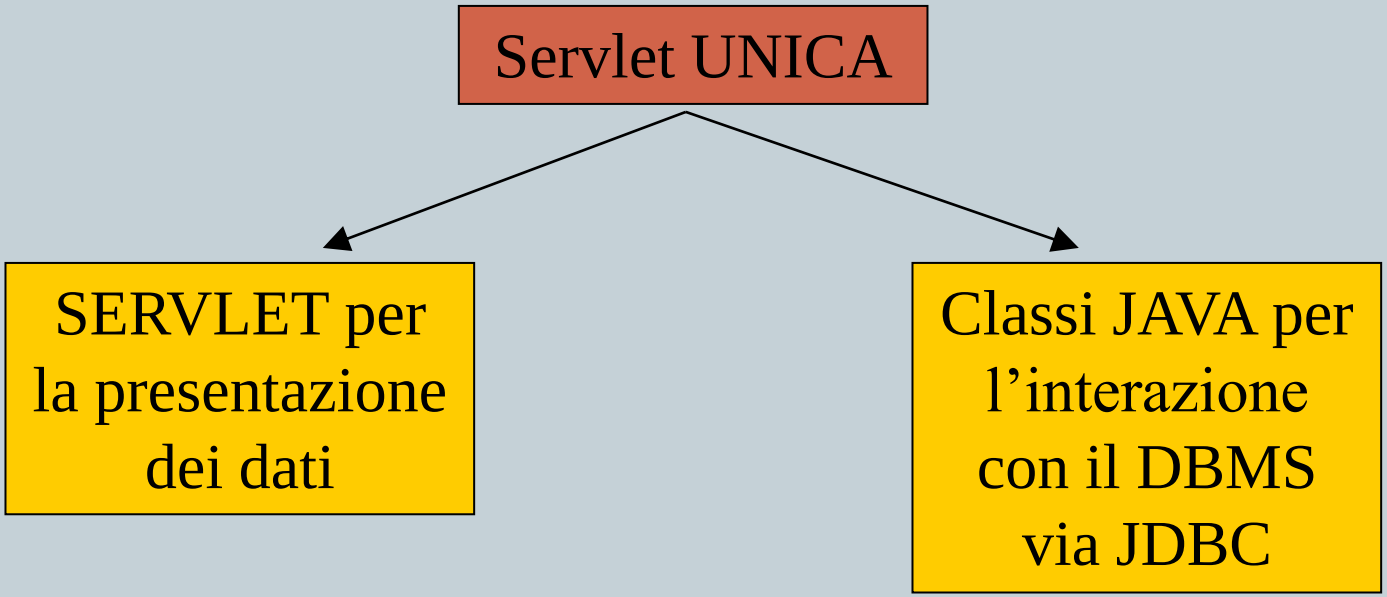
Verso l'architettura MVC-2

Il primo passo verso l'architettura MVC-2 è quello di separare l'interazione con il DBMS dall'elaborazione e presentazione dei dati.



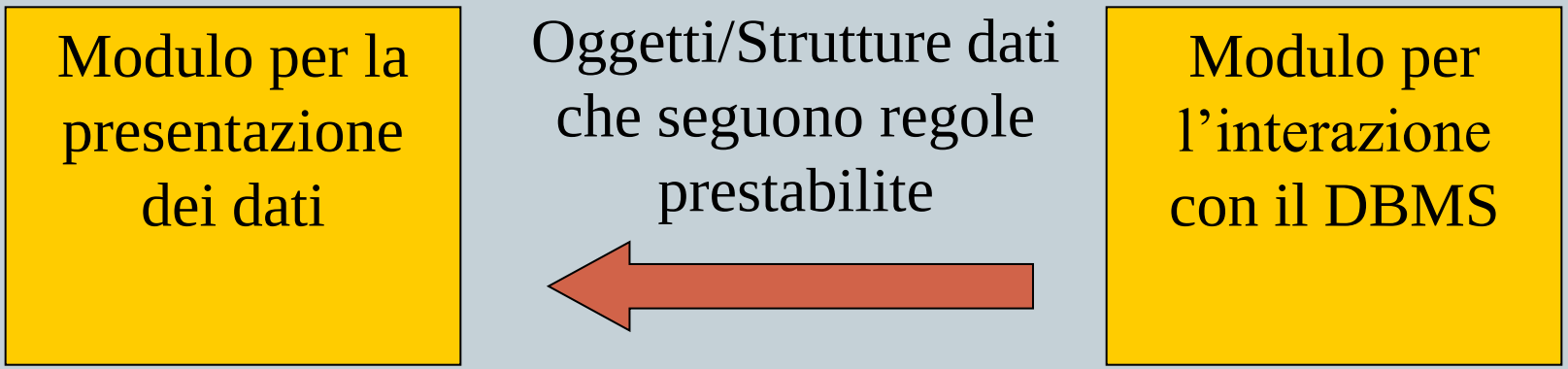
Verso l'architettura MVC-2

Nel contesto JAVA (Servlet + JDBC)



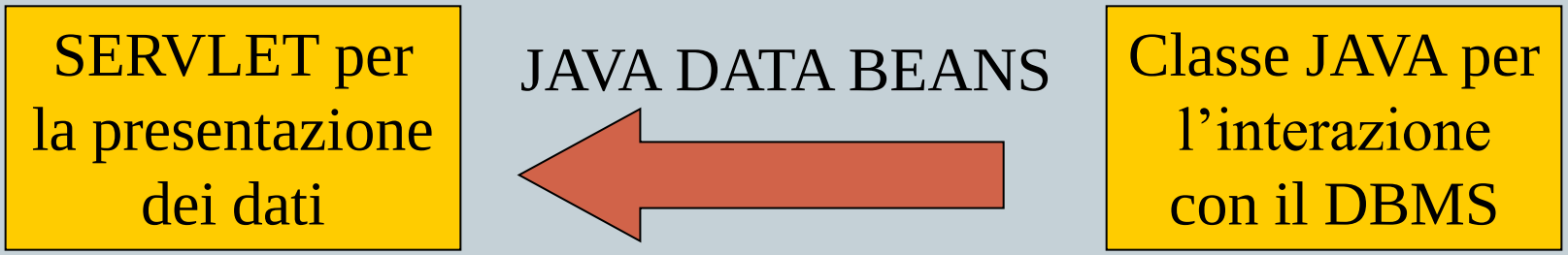
Verso l'architettura MVC-2

Come si trasferiscono i dati estratti dal DBMS alla servlet per l'elaborazione e la presentazione?



Verso l'architettura MVC-2

Nel contesto JAVA (Servlet + JDBC)



JavaBeans

Un Java Bean è un *componente* nella tecnologia Java.

Con il termine *componente* si indicano essenzialmente quelle classi che possono essere utilizzate in modo standard in più applicazioni.

Lo scopo dei componenti infatti è dare la possibilità di *riutilizzare* in modo *sistematico* gli oggetti in contesti diversi, aumentando la produttività.

La definizione di Java Bean è volutamente generica.

Un Java Bean è semplicemente una classe Java che risponde a due requisiti:

- ha un costruttore con zero argomenti;
- soddisfa una serie di direttive relative ai suoi metodi e alle sue variabili.

JavaBeans: metodi “getter”

Una classe Java per essere utilizzata come Java **Data** Bean deve essere scritta seguendo le seguenti direttive:

- **Deve implementare un costruttore senza argomenti.**
  Esempio: PersonaBean();
- **Per ciascuna proprietà della classe che si vuole rendere visibile, deve essere implementato un metodo di nome get*NomeProp*() dove *NomeProp* è il nome della proprietà (con le iniziali maiuscole).**
  Esempio: se una classe Java Bean ha una proprietà numeroTelefono il metodo dovrà essere getNumeroTelefono().

JavaBeans: metodi “setter”

- **Per ciascuna proprietà della classe che si vuole rendere modificabile, deve essere implementato un metodo set*NomeProp*(*ClasseProp* v) dove *NomeProp* è il nome della proprietà (con le iniziali maiuscole) e *ClasseProp* è il tipo della proprietà.**
  Esempio: se una classe Java Bean ha una proprietà numeroTelefono di tipo String, il metodo dovrà essere setNumeroTelefono(String value).

  Inoltre per consuetudine si usa nominare la classe con il suffisso “Bean”.

  Esempio: PersonaBean

JavaBeans nell'interazione con un DBMS

Un Java Data Bean risulta essere il miglior componente per rappresentare

in un oggetto Java una tupla restituita da un'interrogazione SQL e contenuta in un ResultSet

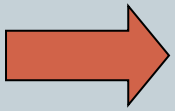
A tal fine, il Bean deve contenere:

1. tante variabili **private** quanti sono gli attributi della tupla;
2. un **costruttore di default** che inizializza agli attributi;
3. i metodi **pubblici** di accesso getter e setter per gli attributi che si vogliono esporre (di solito tutti).

JavaBeans nell'interazione con un DBMS

Persona

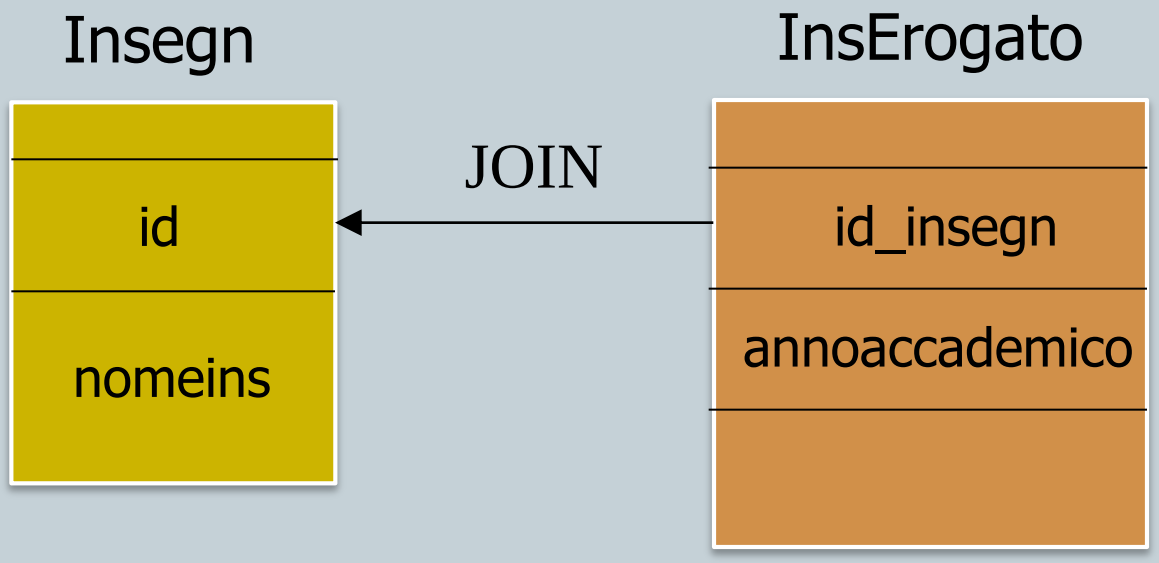
CodiceFiscale	Cognome	Nome	Id	



```
public class PersonaBean {
private String CF, cognome, nome;
private int id;
    PersonaBean() {
    CF = cognome =  nome = "";
    id = -1;
    }
    …
    public String getCF() {
            return CF;
    }
    public void setCF(String c) {
            CF = c;
    }
}
```

Lavorare con i Bean (1)

Non sempre il mapping uno a uno tra gli attributi di una tabella e le proprietà di un bean risulta essere il migliore approccio.
Nel caso di interrogazioni che eseguono join tra più tabelle è conveniente includere nel bean attributi provenienti da più tabelle.
Ad esempio:



Lavorare con i Bean (2)

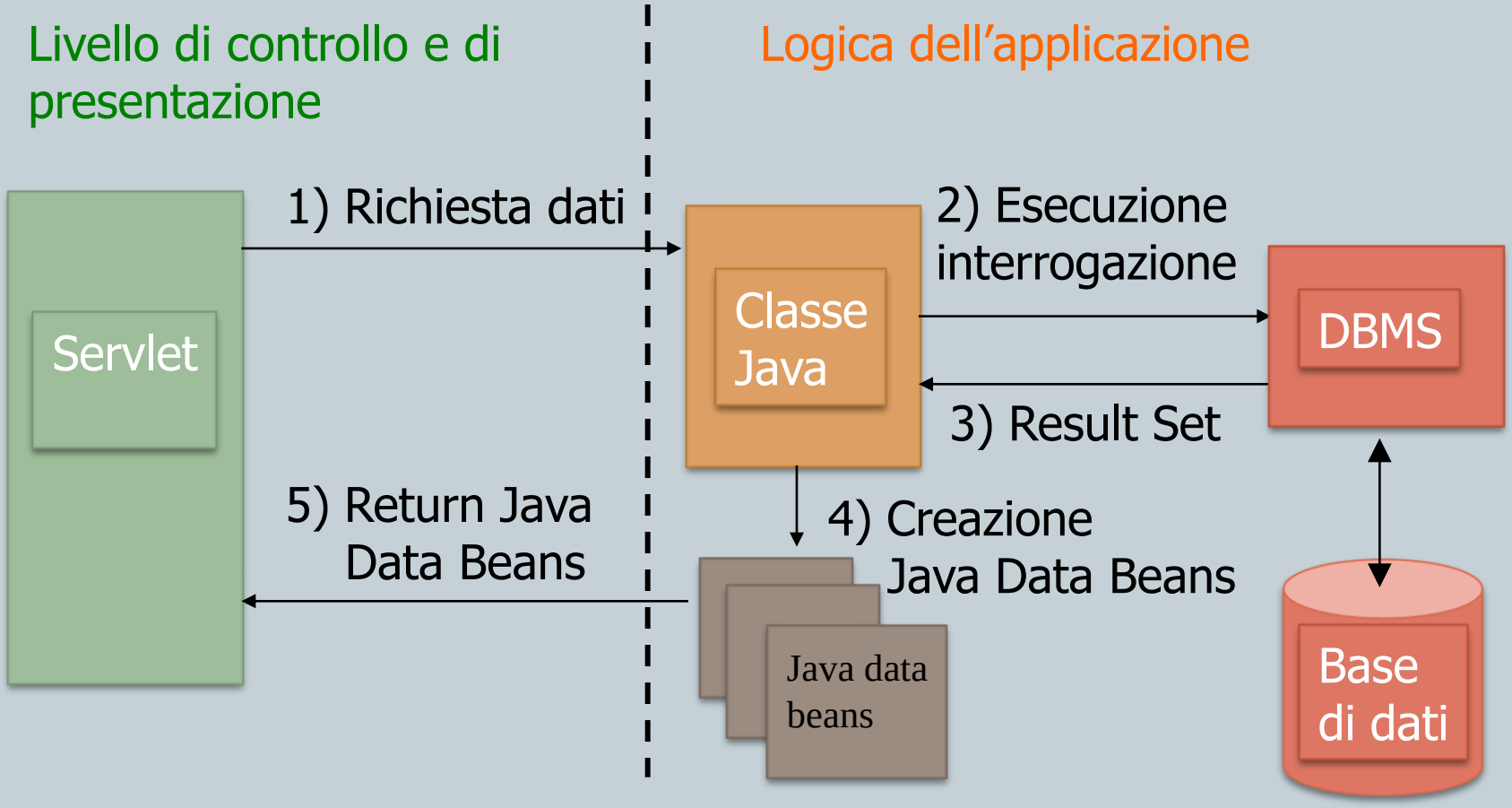
Considerando il caso mostrato nell'esempio del lucido precedente, è necessario:

- Definire una classe Java Bean per rappresentare il risultato dell'interrogazione che esegue un join tra la tabella *InsErogato* e la tabella *Insegn*. Chiamiamo tale classe **InsErogatoBean.**
- E' inoltre consigliabile includere nel bean come proprietà non solo gli attributi della tabella *InsErogato*, ma anche gli attributi di *Insegn* a cui si è interessati;
  ad esempio l'attributo nomeIns può essere incluso e di conseguenza vanno definiti i metodi getNomeIns e setNomeIns nella classe InsEreogatoBean.

Verso l'architettura MVC-2: Java Data Beans

- Parliamo di Java Data Beans quando i Java Beans vengono usati per la gestione del risultato di interrogazioni su una base di dati.
- I Java Data Bean sono quindi dei componenti fondamentali nella strutturazione di una applicazione web centrata sui dati secondo l'architettura MVC-2.
- I Java Data Bean, insieme ad una o più classi Java, permettono di separare la logica dell'applicazione dalla parte di controllo e di presentazione delle informazioni.

Verso l'architettura MVC-2: Java Data Beans



Verso l'architettura MVC-2: Java Data Beans

- La parte logica dell'applicazione è realizzata da un insieme di classi Java e un insieme di Java Data Bean che realizzano le interrogazioni e memorizzano i risultati delle interrogazioni.
  - Ad esempio, in una certa applicazione, tutte le interrogazioni che si possono fare alla base di dati possono essere raggruppate in un'unica classe, che fornirà i Java Data Bean (o Vector di Java Data Bean) ottenuti come risultato di un'interrogazione.
- Il livello di presentazione e di controllo del flusso di esecuzione è realizzato dalle servlet che:
  - analizzano la richiesta HTTP,
  - richiedono alla parte logica i Java Data Bean necessari per formare il documento di risposta e
  - preparano il codice HTML da restituire al browser.

Verso l'architettura MVC-2: Java Data Beans

Assumiamo di chiamare la classe Java che gestisce l'interazione con il DBMS: DBMS.java. Una possibile struttura di questa classe è la seguente:

- Il costruttore di default deve caricare il driver JDBC del DBMS al quale ci si deve connettere.
- Ci devono essere tanti metodi (getXXX o extractXXX) quante sono le interrogazioni che si vogliono gestire. Ogni metodo getXXX deve:
  - Creare un oggetto **connection** per la connessione con il DBMS;
  - Associare eventuali **parametri** d'input con i parametri dell'interrogazione;
  - Eseguire l'**interrogazione**;
  - Creare un **Java Data Bean o un Vector di Java Data Bean** con i dati ottenuti dal DBMS e restituirlo.

Verso l'architettura MVC-2: Java Data Beans

Per strutturare meglio la classe DBMS.java è opportuno creare dei metodi che, dato un **ResultSet**, restituiscono il Java Data Bean (JDB) che rappresenta una riga del **ResultSet**.

Ad esempio nel caso visto in precedenza dovrà essere definito nella classe DBMS il metodo makePersonaBean, che, dato un ResultSet estratto dalla tabella Persona, restituisce un bean contenente i dati della riga corrente del ResultSet.

Verso l'architettura MVC-2: Java Data Beans


```
private PersonaBean makePersonaBean(ResultSet rs)  throws DBMSException {
        bean = new PersonaBean();
        try {
                bean.setId(rs.getInt("id"));
                bean.setCognome(rs.getString("cognome"));
                bean.setNome(rs.getString("nome"));
                bean.setCF(rs.getString("codfiscale"));
                return bean;
        } catch (SQLException e) {
                throw new DBMSException(e.getMessage());
        }
}
```

INTERROGAZIONE SQL:

```
SELECT id, cognome, nome, codfiscale FROM Persona WHERE id=?
```

Java Data Beans: progettazione

Se si decide di rappresentare con un JDB diverso ogni possibile tupla risultato di un'interrogazione, è possibile che il numero di JDB da definire cresca in modo significativo.

Un approccio alternativo consiste nel definire JDB che possono contenere **diverse versioni** (con più o meno attributi) delle tuple ottenute come risultato delle interrogazioni.

Ciò si può ottenere creando inizialmente alcuni JDB che corrispondono alle **principali entità della base di dati** e arricchendo tali JDB con le proprietà necessarie alle varie interrogazioni da eseguire.

Verso l'architettura MVC-2: Java Data Beans

Interazione DBMS e ambiente JAVA con JDB